

# A Task Allocation Schema Based on Response Time Optimization in Cloud Computing

Kai Li

Yong Wang

Meilin Liu

School of Computer Science and Technology,  
Beijing University of Technology, Beijing, China

**Abstract.** Cloud computing is a newly emerging distributed computing which is evolved from Grid computing. Task scheduling is the core research of cloud computing which studies how to allocate the tasks among the physical nodes so that the tasks can get a balanced allocation or each task's execution cost decreases to the minimum or the overall system performance is optimal. Unlike the previous task slices' sequential execution of an independent task in the model of which the target is processing time, we build a model that targets at the response time, in which the task slices are executed in parallel. Then we give its solution with a method based on an improved adjusting entropy function. At last, we design a new task scheduling algorithm. Experimental results show that the response time of our proposed algorithm is much lower than the game-theoretic algorithm and balanced scheduling algorithm and compared with the balanced scheduling algorithm, game-theoretic algorithm is not necessarily superior in parallel although its objective function value is better.

**Keywords:** Cloud Computing, Task Scheduling, Response Time, Optimization, Task Slicing Strategy, Adjustable Entropy Function.

## 1. Introduction

As an Internet-based computing model of public participation, cloud computing[1] is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet. With the rapid development of the Internet, real-time data stream

---

*Correspondence and offprint requests to:* Yong Wang, Pingleyuan 100, Chaoyang District, Beijing, China. e-mail: wangy@bjut.edu.cn

and connected devices' diverse development, SOA's adoption as well as the promotion of search service, social networks, mobile commerce and open collaboration's demands, cloud computing develops rapidly. Currently, Google, IBM, Amazon, Microsoft and other IT vendors are making great efforts to research and promote cloud computing.

The characteristics of cloud computing, such as abstract and virtualized resources, on demand request of the Internet users, isolation for the accesses of different users, dynamic scalability, security under the unsafe networks determine that cloud computing has its own technical architecture as a new distributed computing paradigm. These characteristics will greatly influence the technologies in cloud computing corresponding to those in traditional distributed computing and distributed systems, such as task scheduling technology.

Cloud computing not only provides various business applications or personal applications via the Internet, but also includes the integration of tradition high performance computing under the new environments, for example, Amazon's elastic compute cloud provides high performance computing for the Internet users. Under the large-scale demands for services, especially for the requirements of computing intensive services, how to allocate resources effectively, and how to handle service requests is particularly important for cloud computing platform. There is no doubt that each characteristics of cloud computing will make the task scheduling in cloud computing special, such as the nature and characteristics of powerfully parallel processing capabilities.

In this paper, we introduce a new task scheduling approach based on response time. This approach satisfies the special requirements in cloud computing to some extent and is also novel in traditional distributed computing. We assume that a task can be split into task slices and take the maximum of all task slices' execution time as the response time. When decomposing a task, each scheduler expects its response time minimum. Then we design a task scheduling algorithm. This scheduling method uses response time as the accordance of scheduling decision, and is suitable for cloud computing platforms.

In Section 2, we introduce the related works briefly. We discuss the system structure of a cloud, and introduce a cloud computing system model for computing intensive requirements in Section 3. In Section 4, we establish the mathematical model based on response time and design the task scheduling algorithm. In Section 5, we do a number of detailed experiments which show that this scheduling algorithm is better than the game-theoretic algorithm and the so-called balanced scheduling algorithm. Finally, we conclude this paper in Section 6.

## 2. Related Works

In general, job allocation algorithms in distributed systems can be classified as static or dynamic [2]. In static algorithms, job allocation decisions are made at compile time and remain constant during runtime. For example, in [3], Kim and Kameda proposed a simplified load balancing algorithm, which targets at the minimizing the overall mean job response time via adjusting the each node's load in a distributed computer system that consists of heterogeneous hosts, based on the single-point algorithm originally presented by Tantawi and Towsley. Grosu and Leung[4] formulated a static load balancing problem in single class job distributed systems from the aspect of cooperative game among computers. Also, there exists several studies on static load balancing in multi-class job systems [5, 6]. In contrast, dynamic job allocation algorithms attempt to use the runtime state information to make more informative job allocation decisions. In [7], Delavar introduced a new scheduling algorithm for optimal scheduling of heterogeneous tasks on heterogeneous sources, according to Genetic Algorithm which can reach to better makespan and more efficiency. In [8], Fujimoto proposed a new algorithm RR that uses the criterion called total processor cycle consummation, which is the total number of instructions the grid could compute until the completion time of the schedule, regardless how the speed of each processor varies over time, the consumed computing power can be limited within  $(1 + m(\ln(m-1) + 1)/n)$  ( $m$  represents the number of the processor,  $n$  represents the number of independent coarse-grained tasks with the same length) times the optimal one.

For balanced task scheduling, [9, 10, 11] proposed some models and task scheduling algorithms in distributed system with the market model and game theory. [12, 13] introduced a balanced grid task scheduling model based on non-cooperative game. QoS-based grid job allocation problem is modeled as a cooperative game and the structure of the Nash bargaining solution is given in [2]. In [14], Wei and Vasilakos presented a game theoretic method to schedule dependent computational cloud computing services with time and cost constrained, in which the tasks are divided into subtasks. The above works generally take the scheduler or job manager as the participant of the game, take the total execution time of tasks as the game optimization goals and give the proof of the existence of the Nash equilibrium solution and the solving Nash equilibrium

solution algorithm, or model the task scheduling problem as a cooperative game and give the structure of the cooperative game solution.

In a cloud computing environment, the goal of task scheduling is to achieve the optimal scheduling of jobs submitted by users, and try to improve the overall throughput of the cloud computing system. In recent years, a lot of people have been studying the task scheduling problems in the cloud computing environment and made rich achievements. At present, task scheduling algorithms at home and broad mainly base on the earliest completion time ,quality of service, load balancing, economic principles and so on. Job allocation algorithms can be classified as performance-centric, QoS-centric and economic principle-centric based on the different goals. Performance-centric task scheduling algorithms take the scheduling performance as the ultimate goal such as the shortest completion time, including Max-Min, Min-Min algorithm (e.g.,[15, 16]), genetic algorithm (e.g.[17]), ant colony algorithm (e.g.,[18]). QoS-centric task scheduling algorithms have been studied widely. In [19], Chanhnan and Joshi selected resources based on the weighted average execution time, regarded the network bandwidth as QoS attributes and divided the tasks into two categories , high QoS and low QoS, and high QoS tasks had the priority to be scheduled. In [20], Xu and Wang developed a scheduling strategy for multiple workflows with different QoS requirements. In addition, [21, 22] proposed some task scheduling algorithms from the view of economic principle.

Unlike research works[12-13] directly related to this article in which the task slices are actually executed sequentially, the task slices generated by the scheduler are executed in parallel here.

According to the classification of static algorithms or dynamic ones for task scheduling algorithms, our algorithm can be classified into a semi-dynamic algorithm, just because this algorithm utilizes the states and capacity of computing nodes in a statistic way.

This paper has two main contributions as follows:

1. Based on the cloud computing system model, we establish the mathematical model based on the response time.
2. In view of difficulty of the above mathematical model for solving optimization problems, based on adjusting maximum entropy method, we give an approximate solution of the optimization model and design a new task scheduling algorithm. From the experiments, we can see that this scheduling algorithm has better optimal results than the game-theoretic algorithm and balanced scheduling algorithm.

### 3. System Model for A Cloud System

A variety of services in the cloud computing platform can be roughly grouped into two categories: data-compute-intensive services and interaction-intensive services [23]. The former have a higher complexity and requires a higher computing power; the latter can be classified as general Web Services which needs higher real-time requirements. Fig.1 is a cloud system simulation based on the two type of services. Firstly, the service request will be classified to the Web Service queue or HPC (High performance computing) service queue according to its type, and then calls cloud resources depending on the type of service.

It is well known that the architecture of a cloud system roughly includes application layer, platform layer, virtual resource layer and physical layer [1] shown as Fig.2. The application layer accepts the requests of the Internet users through various applications, then deliver the requests to the platform layer. The platform layer usually contains a task scheduler to split the user tasks into pieces and deliver these task pieces onto different virtual resources. There are mappings from virtual resources to physical resources using the so-called virtualization technology.

In addition, data-compute-intensive services are usually very complex, and for these service requests, a cloud system should give full play to their own advantages of task parallel processing and the following three assumptions should be reasonable:

1. In the cloud computing system model, the scheduler can split the task into task slices. Because the configuration of a node is controllable within a mesh range, we can assume that all the computing nodes can handle the task slices;
2. The internal processing cost of the scheduler can be ignored, namely, we can assume that the task slices' transmission cost on the network and execution cost on the computing nodes are the key consideration of the task execution cost;

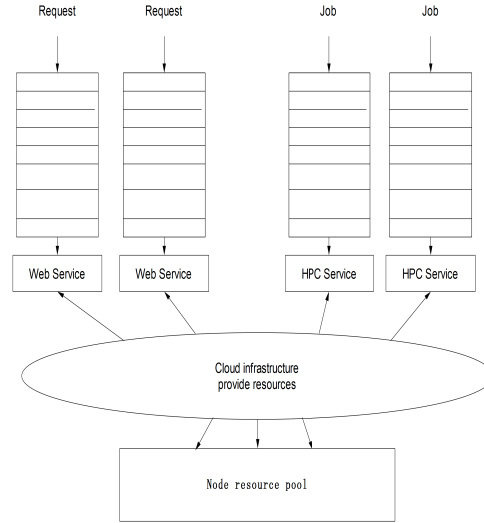


Fig. 1. Cloud System Simulation

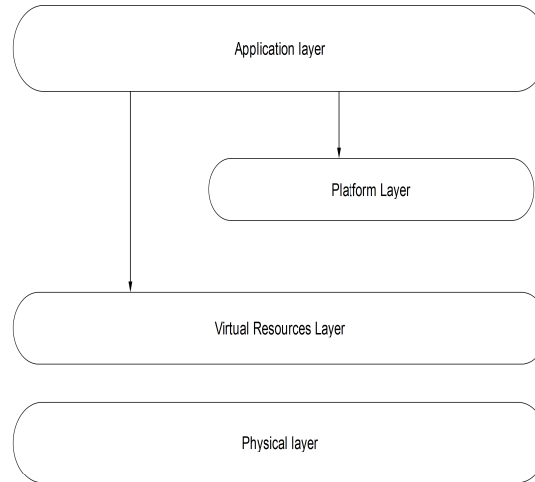


Fig. 2. Cloud System Architecture

3. In the cloud computing system, node resources that provide service may be subjected to the M/G/1 queueing system.

Conjunction with Fig.1 and Fig.2, the system model of a cloud system for compute-intensive requirements shown in Fig.3 is reasonable. In this system model, the number of computing nodes that can provide service is assumed to be  $m$ , the number of users that generate service request is  $l$ , and the number of schedulers that allocates resource to implement the service request is  $n$ .

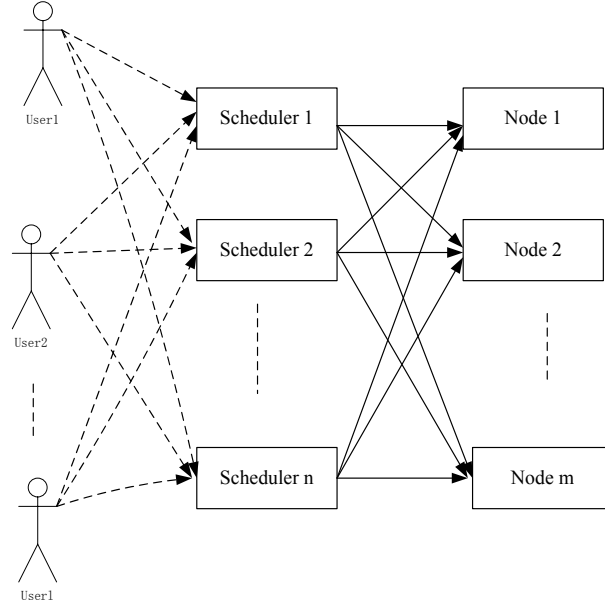


Fig. 3. System Model

- User: a user generates a request(task) to a scheduler and user  $k$  is assumed to generate jobs with average rate  $\beta_k$ (jobs per second ) according to a Poisson process independently. Jobs are then sent by the user to a scheduler that dispatches them to the computing nodes.
- Scheduler: a scheduler receives job from a set of users and then assigns them to computing nodes in the cloud computing system. According to the first assumption, the task decomposition cost is negligible.
- Task slice: depending on the number of computing nodes, scheduler  $i$  dispatches the users' tasks into  $m$  task slices,  $a_{ij}$  is the ratio of scheduler  $i$  assigns one task to the computing node  $j$  which satisfies the following constraint:

$$a_{ij} \geq 0 \quad \text{and} \quad \sum_{j=1}^m a_{ij} = 1 \quad (1)$$

- Computing node: the computing node executes and processes task slices sent to it. According to the second assumption, the computing node has the ability to execute the general task slice.  $\mu_j$  is the average processing rate of jobs at computational node  $j$ , the execution time can be subjected to arbitrary distribution, and each computing node can be modeled as an M/G/1 queuing system. For stability, we have two constraints as follows:

$$\sum_{i=1}^n \lambda_i < \sum_{j=1}^m \mu_j \quad (2)$$

$$\sum_{i=1}^n \lambda_i a_{ij} < \mu_j \quad (3)$$

Where  $\lambda_i$  is the average rate of jobs that scheduler  $i$  issues. The first constraint means that the average rate of jobs that all schedulers issues must not be faster than average rate of jobs that all computing nodes execute. The second constraint means that the average rate of jobs sent to computing node  $j$  must not exceed the rate at which jobs can be executed by the computing node  $j$ .

## 4. Mathematical Model to Optimize the Task Response Time

### 4.1. Objective Function

Let  $a_i = \{a_{i1}, a_{i2}, \dots, a_{im}\}$  represents scheduler  $i$ 's task slicing scheme on all cloud computing nodes, where  $i = 1, 2, \dots, n$  and  $a_{ij}$  is the task's ratio of jobs according which scheduler  $i$  assigns jobs to computing node  $j$ . We define the maximum processing time among the task slices as this task's response time. Each scheduler expects that its task response time will be the shortest, thus, we can establish the objective function based on the task response time. From the above assumption two, we can know that task slice's processing time contains two parts: the transmission time from a scheduler to a computing node and the execution time at the computing node. Assume that the average length of all tasks is  $b$  bits, the transmission delay time from scheduler  $i$  to the computing node  $j$  is  $e_{ij}$ , the communication bandwidth from scheduler  $i$  to computing node  $j$  is  $c_{ij}$ , the transmission time of task slice  $a_{ij}$  can be calculated by formula 4.

$$L_{ij} = e_{ij} + \frac{b \times a_{ij}}{c_{ij}} \quad (4)$$

According to reference[24], the average service time for the M/G/1 queuing system is as follows:

$$E(t) = \frac{1}{\mu} + \frac{\lambda(\sigma^2 + \frac{1}{\mu^2})}{2(1 - \frac{\lambda}{\mu})} \quad (5)$$

Where  $\sigma^2$  is the variance of the service time,  $\mu$  is the average execution rate of the computing node. Assume that the computing node's service time subjects to the negative exponential distribution, thus the average service time  $F_{ij}$  of task slice  $a_{ij}$  on the computing node is shown as follows:

$$F_{ij} = (\frac{1}{\mu_j} + \frac{\sum_{k=1}^n \lambda_k a_{kj}}{\mu_j(\mu_j - \sum_{k=1}^n \lambda_k a_{kj})}) \times a_{ij} \quad (6)$$

The execution time of task slice  $a_{ij}$  is the sum of the transmission time  $L_{ij}$  and the average service time  $F_{ij}$  on the computing node  $j$ , namely:

$$F_{ij} + L_{ij} = (\frac{1}{\mu_j} + \frac{\sum_{k=1}^n \lambda_k a_{kj}}{\mu_j(\mu_j - \sum_{k=1}^n \lambda_k a_{kj})}) \times a_{ij} + e_{ij} + \frac{b \times a_{ij}}{c_{ij}} \quad (7)$$

After the slices of a task are scheduled to the computing nodes, each slice is executed independently. the response time of this task is:

$$\begin{aligned} FL_i(a_i) &= \max_{j=1}^m (F_{ij} + L_{ij}) \\ &= \max_{j=1}^m ((\frac{1}{\mu_j} + \frac{\sum_{k=1}^n \lambda_k a_{kj}}{\mu_j(\mu_j - \sum_{k=1}^n \lambda_k a_{kj})}) \times a_{ij} + e_{ij} + \frac{b \times a_{ij}}{c_{ij}}) \end{aligned} \quad (8)$$

Each scheduler when scheduling the task, always expects a minimum response time, and therefore we can get the objective function of scheduler  $i$  based on response time optimization:

$$D_i = \min_{j=1}^m ((\frac{1}{\mu_j} + \frac{\sum_{k=1}^n \lambda_k a_{kj}}{\mu_j(\mu_j - \sum_{k=1}^n \lambda_k a_{kj})}) \times a_{ij} + e_{ij} + \frac{b \times a_{ij}}{c_{ij}}) \quad (9)$$

In order to facilitate the algorithm description, we introduce a new variable  $\mu_{ji}$ , shown in (10).  $\mu_{ji}$  is defined as the computational power of computing node  $j$  that is available to scheduler  $i$  and can be estimated for each computing node  $j$ :

$$\mu_{ji} = \mu_j - \sum_{k=1, k \neq i}^n \lambda_k a_{kj} \quad (10)$$

Using(10), (9) becomes:

$$D_i = \min_{j=1}^m ((1 + \frac{\mu_j - \mu_{ji} + \lambda_i a_{ij}}{\mu_{ji} - \lambda_i a_{ij}}) \times \frac{a_{ij}}{\mu_j} + e_{ij} + \frac{b \times a_{ij}}{c_{ij}}) \quad (11)$$

(7) is convex, which can be proven as follows:

$$\frac{\partial(F_{ij} + L_{ij})}{\partial a_{ij}} = \frac{\mu_{ji}}{(\mu_{ji} - \lambda_i a_{ij})^2} + \frac{b}{c_{ij}} > 0$$

$$\frac{\partial^2(F_{ij} + L_{ij})}{\partial a_{ij}^2} = \frac{2\lambda_i \mu_{ji}}{(\mu_{ji} - \lambda_i a_{ij})^3} > 0$$

## 4.2. Adjusting Entropy Function Method

When  $m \geq 2$ , (11) is a non-differentiable optimization problem and it is difficult to get its solution. [25] introduces a maximum entropy method to solve the unconstrained minimax problem:

$$\min \max_{i=1}^m f_i(x) \quad (12)$$

where  $f_i(x)$  is a continuously differentiable function in  $R^n$ .

The maximum entropy function is shown in (13).

$$FL^p(x) = \frac{1}{p} \ln \left\{ \sum_{i=1}^m \exp(p f_i(x)) \right\} \quad (13)$$

When  $p \rightarrow \infty$ ,  $FL^p(x)$  uniformly converges to

$$\max_{i=1}^m f_i(x)$$

. Because this method simply increases the variable  $p$ , it is very difficult to program practically when  $p$  is very large. In order to solve this problem, [26] introduces an improved entropy function shown in (14) and proposes an adjustable function algorithm.

$$F_p(x, \mu) = \frac{1}{p} \ln \left| \sum_{i=1}^m \mu_i \exp\{p f_i(x)\} \right| \quad (14)$$

The variable parameter  $\mu$  can be calculated by (15).

$$\mu_i^{(k+1)} = \frac{\mu_i^k \exp\{p^{(k)} f_i(x)\}}{\sum_{j=1}^m \mu_j^k \exp\{p^{(k)} f_j(x)\}} \quad (15)$$

where  $\mu_i^{(0)} = \{\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\}$ ,  $i = 1, 2, \dots, m$ .

The adjustable function algorithm which is used to solve the minimax problem (12) is shown as follows:

- Step 1. Given a sufficiently large value  $P > 0$ ,  $\varepsilon > 0$ ,  $p^{(0)} > 0$ ,  $\mu_i^{(0)} > 0$ ,  $r > 1$ ,  $k = 0$ , the initial point  $x^{(0)}$ ;
- Step 2. Solve  $\min F_{p^{(k)}}(x, \mu^{(k)})$  with  $x^{(k)}$  as the starting point to get the new point  $x^{(k+1)}$ . if  $\|x^{(k+1)} - x^{(k)}\|_2 < \varepsilon$ , stop, else go on;
- Step 3. Calculate  $\mu_i^{(k+1)}$  with (15),  $i = 1, 2, \dots, m$ . if  $p^{(k)} < P$ ,  $p^{(k+1)} = r p^{(k)}$ , else  $p^{(k+1)} = p^{(k)}$ .  $k = k + 1$ , goto step 2;

This algorithm is very effective by adjusting  $p$  and  $\mu$ , avoiding  $p$  being too large.

## 4.3. Task scheduling algorithm

For the objective function (9), we can design an approximate algorithm on the basis of the above method as the following process: each scheduler calculates a best task slicing scheme  $a_i$  that results in minimum  $D_i$  in cycles until all schedulers' task slicing schema  $a$  reaches an equilibrium or the cycles reach a certain value. In this paper,  $f_i(x)$  is replaced by  $F_{ij} + L_{ij}$ .

On the basis of the above analysis, we obtain the task scheduling algorithm as follows:

- Step 1. System parameters initialization: Let  $n$  represents the number of schedulers in the cloud computing system,  $m$  represents the number of the computing nodes,  $\lambda_i(0)$  is the average rate of jobs that scheduler  $i$  issues,  $\mu_j(0)$  is the average processing rate of jobs at computing node  $j$ , the transmission delay time from scheduler  $i$  to computing node  $j$  is  $e_{ij}(0)$ , the average data length of all tasks is  $b(0)$  bits, the communication bandwidth from scheduler  $i$  to computing node  $j$  is  $c_{ij}(0)$  Kbps, where  $i = 1, 2, \dots, n, j = 1, 2, \dots, m$ . The task slice program of scheduler  $i$  is initialized as:  $a_i(0) = \{a_{i1}(0), a_{i2}(0), \dots, a_{im}(0)\} = \{\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\}$ ,  $maxCycle(0)$  is the maximum cycle number, the current cycle  $currentCycle$  is 1, the maximum adjusting value is  $P(0)$ ,  $formerA = latterA = a(0)$ , the initial value of  $formerA$  and  $latterA$ 's norm difference  $diffA$  is 1, the error precision is  $\varepsilon(0) = 10^{-4}$ ;
- Step 2. Based on the above initial values, calculate  $\mu_{ji}$  by (10);
- Step 3. if  $diffA > \varepsilon$  and  $currentCycle < maxCycle$ , then the task scheduling program is calculated and the cycle ends; otherwise, repeat the following step 4 to step 5;
- Step 4: Perform the following steps from  $i = 1$  to  $n$ ;
  - Step 4.1. Let  $x0 = latterA_i$ , where  $latterA_i$  represents the  $i$ th row of  $latterA$ ,  $\mu^{(0)} = \{\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\}$ ,  $p = 10$ ,  $r = 10$ ,  $k = 0$ ,  $formerX = latterX = x0$ ;
  - Step 4.2. Solve  $minF_{p^{(k)}}(x, \mu^{(k)})$  with  $latterX$  as  $x$ 's starting point to get the new point  $x$ ;
  - Step 4.3. Let  $formerX = latterX$ ,  $latterX = x$ . if  $\|formerX - latterX\|_2 < \varepsilon$ , goto step 4.5, else go on;
  - Step 4.4. Calculate  $\mu_i^{(k+1)}$  with (15),  $i = 1, 2, \dots, m$ ,  $k = k + 1$ . if  $p^{(k)} < P$ ,  $p^{(k+1)} = rp^{(k)}$ , else  $p^{(k+1)} = p^{(k)}$ . Goto step 4.2;
  - Step 4.5.  $latterA_i = latterX$ , calculate  $\mu_{ji}$  with the new  $latterA$ ;
- Step 5. Let  $diffA = \|latterA - formerA\|_2$ ,  $formerA = latterA$ , goto step 3.

The above task scheduling algorithm determines the task slicing program according to which each scheduler should allocate jobs to each computing node in order to minimize all schedulers' response time.

## 5. Experiments

In this section, we do a series of experiments to analyze the merits and demerits of this schema by comparison with other algorithms, such as game-theoretic algorithms [13] and the balanced scheduling algorithm [27]. The proposed algorithm is labeled as PS. The game-theoretic algorithm uses the game theory to design a new task scheduling algorithm with goal of completion time, which is labeled GS. The balanced scheduling algorithm allocates tasks to processors in proportion to its computing power(task processing rate), which is labeled BS; the faster computing node are sent more tasks by the schedulers. The proportion of tasks sent to the computing nodes is given by the following:

$$a_{ij} = \frac{\mu_{ji}}{\sum_{j=1}^m \mu_{ji}} \quad (16)$$

Let  $\phi_i$  represents the relative job arrival rate of scheduler  $i$ , the average arrival rate of scheduler  $i$  is calculated under the below formula.

$$\lambda_i = \phi_i \cdot \rho \cdot \sum_{j=1}^m \mu_j \quad (17)$$

Where  $\rho$  is the required overall average system load.

### 5.1. Objective Function Value and Response Time Value

In a cloud computing system, the processing abilities of the computing nodes may be have little difference or not. Considering these two cases, we do two sets of experiments. In each case, we compare the objective



Table 1. Relative Job Arrival Rate of Each Scheduler

scheduler	1	2-5	6	7
Relative job arrival rate	0.0035	0.01	0.006	0.005

Table 2. Average Task Processing Rate of the Computing Nodes

Computing node	1	2	3	4
Average Task Processing Rate	0.28	0.22	0.19	0.23
Computing node	5	6	7	8
Average Task Processing Rate	0.20	0.26	0.22	0.23

function values and response time values of the above three different task scheduling algorithms. Firstly, we set the overall average system load to 50 percent ( $\rho = 0.5$ ), the average length of all tasks is 1Mbits ( $b = 1\text{Mbits}$ ), the average transmission delay time from scheduler  $i$  to computing node  $j$  is 0.5second ( $e_{ij} = 0.5\text{s}$ ), the communication bandwidth from scheduler  $i$  to computing node  $j$  is 100Kbps. At the same time, we assume that this cloud computing system has  $n = 7$  schedulers and  $m = 8$  computing nodes. The relative job arrival rate of each scheduler is shown at Table 1.

In the first set of experiment, there are some computing nodes that have higher average processing rate than the others clearly and the average processing rate of each computing nodes is shown at Table 2.

Under the above initial condition, the objective function value of each scheduler with different task scheduling algorithm is shown in Fig 4 shows.

As can be seen from Fig.4, when the computing abilities of each computing node are different, the objective function value of each scheduler of our proposed task scheduling algorithm is significantly better

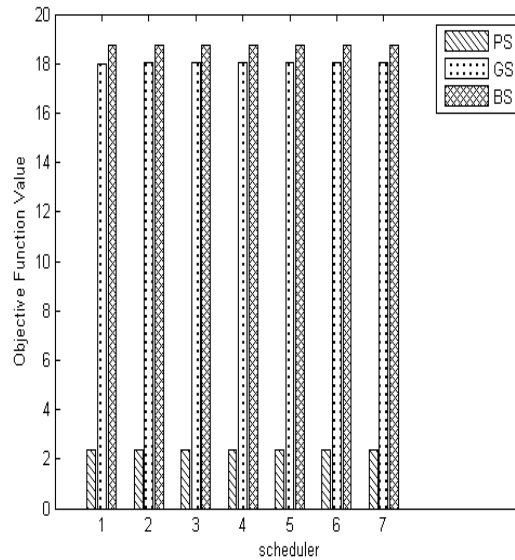


Fig. 4. Objective Function Value of Each Scheduler

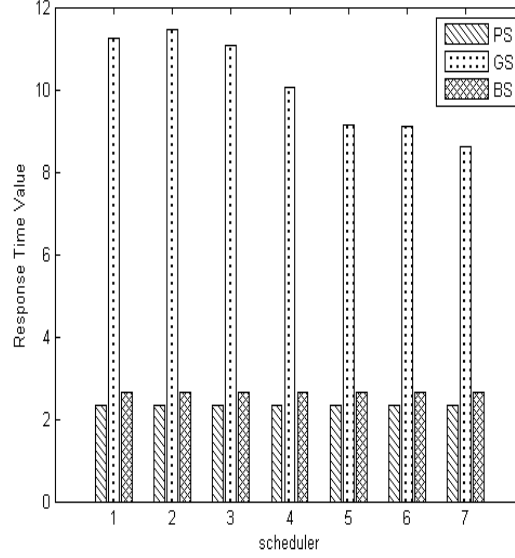


Fig. 5. Response Time Value of Each Scheduler

Table 3. Average Task Processing Rate of the Computing Nodes

Computing node	1	2	3	4
Average Task Processing Rate	0.25	0.26	0.23	0.24
Computing node	5	6	7	8
Average Task Processing Rate	0.22	0.25	0.22	0.23

than the other two algorithms. The game-theoretic algorithm is slightly better than the balanced scheduling algorithm. The objective function values among the schedulers are little different in each task scheduling algorithm. In order to study the effects of GS and BS in the implementation of task slices executed in parallel, we apply these two algorithms to the response time function and get each scheduler's response time shown in Fig.5.

From Fig.5, each scheduler's response time of PS is slightly better than BS, and significantly better than GS. In addition, the response time values of PS and BS are more balanced than GS.

In the second set of experiment, the processing abilities of all computing nodes have little difference, which are shown in Table 3.

Under the above initial condition, the objective function value and the response time value of each scheduler with different task scheduling algorithm are respectively shown in Fig.6 and Fig.7.

From Fig.6 and Fig.7, we can see that there are similar results whether the processing abilities of each computing node are balanced or not by comparison with the first experiment. Combining these two sets of experiments, whether the cloud computing system provides computing nodes' processing abilities balanced or not, we can draw the following conclusions:

**Conclusion 1:** the response time values among the schedulers under PS and BS are balanced but for GS;

**Conclusion 2:** Both from response time value and the objective function value, the proposed task scheduling algorithm is superior to the other two algorithms;

**Conclusion 3:** Compared with BS, GS is not necessarily superior in parallel although its objective function value is better.

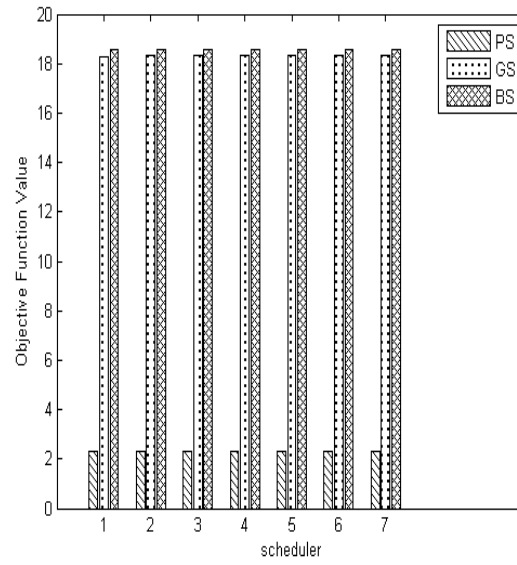


Fig. 6. Objective Function Value of Each Scheduler

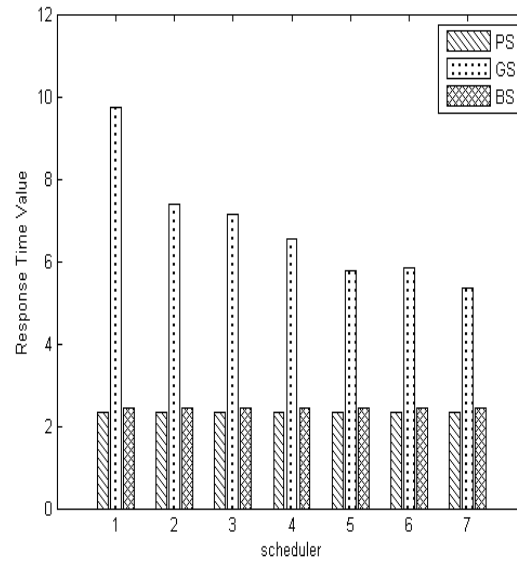


Fig. 7. Response Time Value of Each Scheduler

## 5.2. Effects of System Load

In this section, we compare the objective function values of the above three task scheduling algorithms when schedulers' arrival tasks increase. Here, we vary the average load of the system from 0.1 to 0.9. The remaining conditions are the same as the second set of experiment in the preceding experiments.

From the preceding experiments we can see that all scheduler's objective function values are similar under each task scheduling algorithms whether the processing abilities that the computing nodes provide

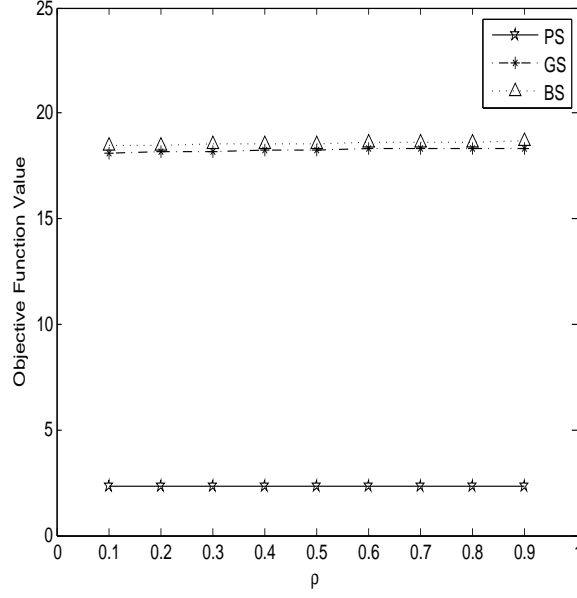


Fig. 8. Objective Function Value versus System Load

are balanced or not. Therefore, we use the first scheduler's objective function value to compare the variation over system load in different algorithms. But the response time values among all schedulers are significantly different. In order to verify the merits of our proposed algorithm in response time, we compare the maximum of all schedulers' response time value under PS with the minimum of all schedulers' response time values under GS and BS. If the maximum value of PS is lower than the minimum of GS and BS, it is obvious that we can draw the conclusion that our algorithm is better than the others for the whole system.

Under the above initial conditions, we calculate the first scheduler's objective function values over the system load from 0.1 to 0.9, Fig.8 shows the results.

From Fig.8, we can find that the objective function values under three algorithms is stable and the objective function value under PS is lower than GS and BS.

Fig.9 shows the maximum of all schedulers' response time values under PS and the minimum of all schedulers' response time value under GS and BS with the system load varying from 0.1 to 0.9.

From Fig.9 we can find that: when the system load increases, although the minimum of all schedulers' response time value under GS decreases, PS is still better than GS and BS all the time.

With Fig.8 and Fig.9, when the system load increases, our proposed algorithm is still better than the other two algorithms.

### 5.3. Effects of System Size

In this part of the experiment, system size mainly refers to the number of the schedulers and the number of computing nodes, and we also do two sets of experiments separately.

In the first set of experiment, we consider the effect of the number of the schedulers. We vary the number of the schedulers from 7 to 15, and the number of the computing nodes is 10. Table 4 shows ten computing nodes' average task processing rates and table 5 shows fifteen schedulers' relative job arrival rates. The remaining conditions are the same as the first experiment.

Under the above initial conditions, the first scheduler's objective function values are shown in Fig.10 and the maximum of all schedulers' response time value under PS with the minimum of all schedulers' response

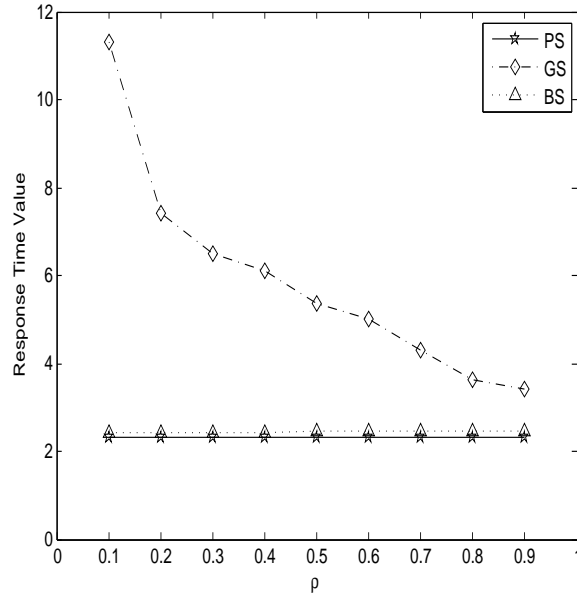


Fig. 9. Response Time Value versus System Load

Table 4. Average Task Processing Rate of the Computing Nodes

Computing node	1	2	3	4-5
Average Task Processing Rate	0.25	0.26	0.23	0.23
Computing node	6	7	8-9	10
Average Task Processing Rate	0.21	0.24	0.24	0.22

time values under GS and BS are shown in Fig.11 when the cloud computing system adds a scheduler each time

Fig.10 and Fig.11 show that our tasking scheduling algorithm is better than the other two algorithms when the number of schedulers increases.

In the second set of experiment, we investigate the effect of the number of computing nodes. We vary the number of computing nodes from 10 to 15 and set the number of the scheduler to 7. The average task processing rate and the average relative job arrival rate are listed in table 6 and table7 separately. The remaining conditions are the same as the first experiment.

Under the above conditions, the first scheduler's objective function values are shown in Fig.12 and the maximum of all schedulers' response time value under PS with the minimum of all schedulers' response time

Table 5. Relative Job Arrival Rate of Each Scheduler

scheduler	1	2-5	6	7
Relative job arrival rate	0.0035	0.01	0.006	0.005
scheduler	8-10	11-13	14	15
Relative job arrival rate	0.003	0.002	0.0015	0.0015

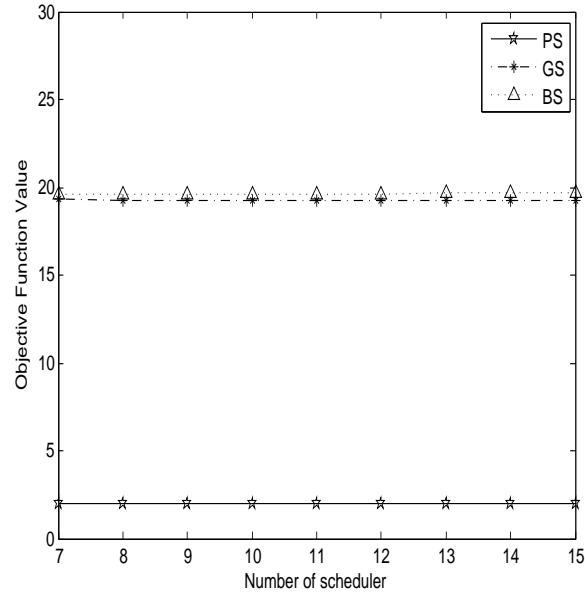


Fig. 10. Objective Function Value versus the Number of Scheduler

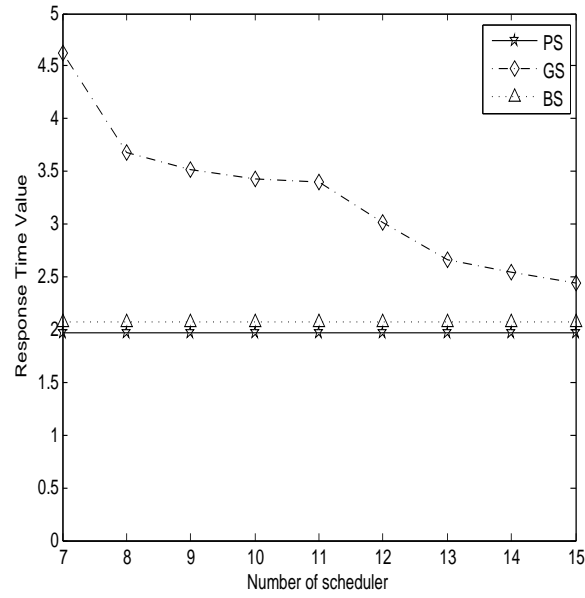


Fig. 11. Response Time Value versus the Number of Scheduler

Table 6. Average Task Processing Rate of the Computing Nodes

Computing node	1	2	3-5	6
Average Task Processing Rate	0.25	0.26	0.23	0.21
Computing node	7-9	10-13	14	15
Average Task Processing Rate	0.24	0.22	0.20	0.20

Table 7. Relative Job Arrival Rate of Each Scheduler

scheduler	1	2-5	6	7
Relative job arrival rate	0.0035	0.01	0.006	0.005

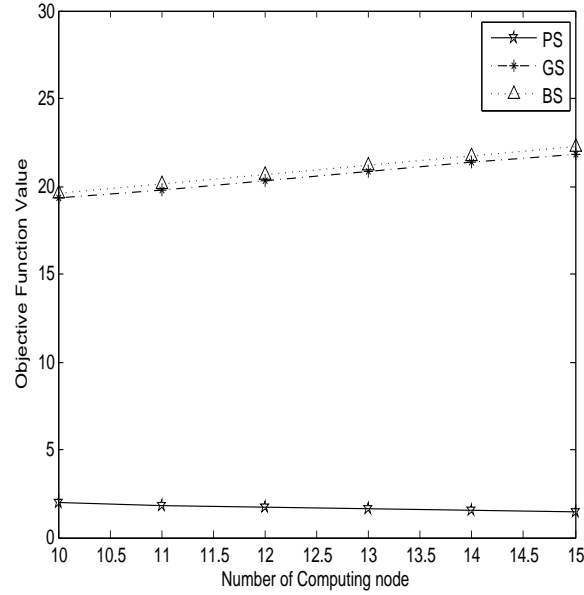


Fig. 12. Objective Function Value versus the Number of Computing Node

values under GS and BS are shown in Fig.13 when the cloud computing system adds a computing node each time.

As can be seen from Fig.12 and Fig.13, our tasking scheduling algorithm is better than the other two algorithms when the number of computing nodes increases.

According to the above two sets of experiments, we can draw this conclusion: when the system size increases, PS is still better than the other two algorithms.

#### 5.4. Effects of Communication Bandwidth

In the previous experiments, we assume that the communication bandwidth is const. In fact, by using the new transmission medium, the communication bandwidth gets faster and faster, which will affect the task scheduling algorithm. In this set of experiment, we will study its effects on the response time and objective function value when the communication bandwidth is 100, 500 and 1024 Kbps. The remaining conditions are the same as the second set of experiment in the first experiment. Fig.14 shows the each scheduler's objective function values when the communication bandwidth changes under different task scheduling algorithm.

From Fig.14, we can see that when the communication bandwidth increases, each scheduler's objective function value decreases; each scheduler's objective function value are little difference under each algorithm; PS is still significantly better than GS and GS is a little better than BS.

Each scheduler's response time values when the communication bandwidth changes under different task scheduling algorithm are shown in Fig.15.

As can be seen from Fig.15, with the communication bandwidth increasing, each scheduler's response time value decrease; each scheduler's response time value has little difference under PS and BS, but for GS.

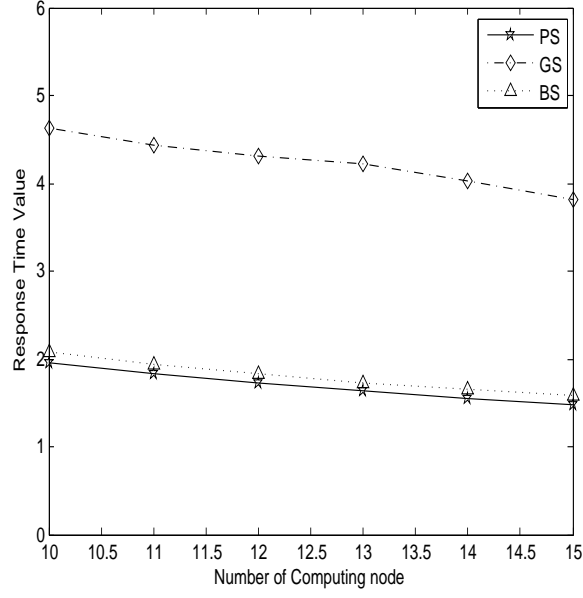


Fig. 13. Response Time Value versus the Number of Computing Node

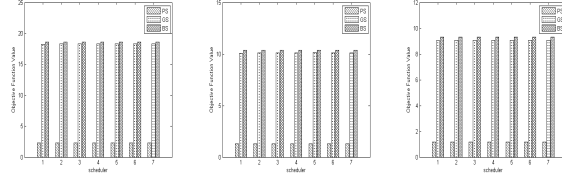


Fig. 14. Each Scheduler's Objective Function Value versus Communication Bandwidth

### 5.5. Fairness

In this part of the experiment, we investigate the fairness of each algorithm. When the average response time value for each scheduler is the same, the fairness is achieved. If one scheduler's response time value is lower and another's is higher, then the scheduling algorithm can be considered unfair. A fairness index[2] is given by

$$FI = \frac{(\sum_{i=1}^n D_i)^2}{n \sum_{i=1}^n D_i^2} \quad (18)$$

Where  $D_i$  is the average response time of scheduler  $i$ . When a scheduling algorithm's fairness index is

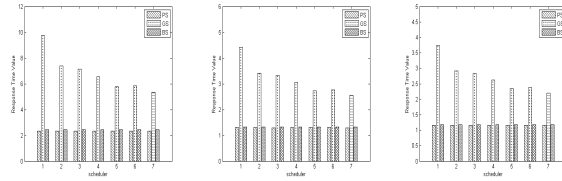


Fig. 15. Each Scheduler's Response Time Value versus Communication Bandwidth



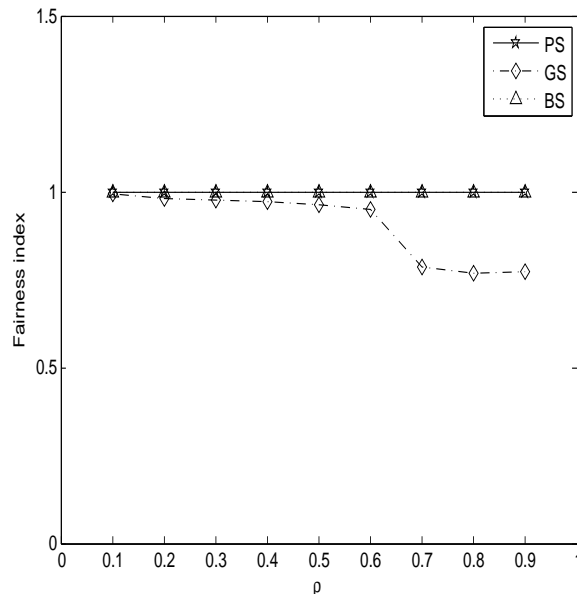


Fig. 16. Fairness versus System Load

closer to 1.0, it is more fair. Here, we study the above three scheduling algorithm's fairness under different conditions such as system load, system size. Their initial conditions are the same as the above.

Fig.16 shows the fairness index when the system load varies from 0.1 to 0.9 under different algorithms. From this figure, we can find that the fairness indexes of both PS and BS are very closer to 1, which means that PS and BS are fair. But for GS, when the system load is low, the fairness is high. However, the fairness goes lower when the system load gets higher.

When the number of scheduler changes, the fairness indexes of different algorithms are shown in Fig.17. Fig.18 shows the fairness indexes when the number of computing node changes. From Fig.17 and Fig.18, we can find that the fairness indexes are close to 1.0 all the time for PS and BS; but for GS, when the system size increases, its fairness index decreases.

From the above three experiments, we can draw this conclusion: our proposed algorithm is fair.

## 6. Conclusions

This paper discussed a new task scheduling algorithm about response time. On the premise that a task can be broken down into task slices, we build a mathematical model in which task slices are actually executed in parallel and design a new task allocation schema. From the above experiments, we can draw these conclusions: our schema is fair, stable and has a significantly advantage than the game-theoretic algorithm and balanced scheduling algorithm.

## References

- [1] I. FOSTER, Y. ZHAO, I. RAICYI, et al. *Cloud Computing and Grid Computing 360-degree Compared*, In: Proc. of the 2008 Grid Computing Environments Workshop(GCE2008), 1-10.
- [2] R. SUBRATA, A. Y. ZOMAYA, B. A. LANDRELDT. *A Cooperative Game FrameWork for QoS Guided Job Allocation Schemes in Grids*, IEEE Transactions on Computers, 2008, 57(10):1413-1422.
- [3] C. KIM, H. KAMEDA. *An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems*, IEEE Transactions on computers, 1992, 41(3):381-384.
- [4] D. GROU, M. LEUNG. *Load Balancing in Distributed Systems: An Approach Using Cooperative Games*, In: Proc. of the International Parallel and Distributed Processing Symposium, 2002, 1530-2075/02.

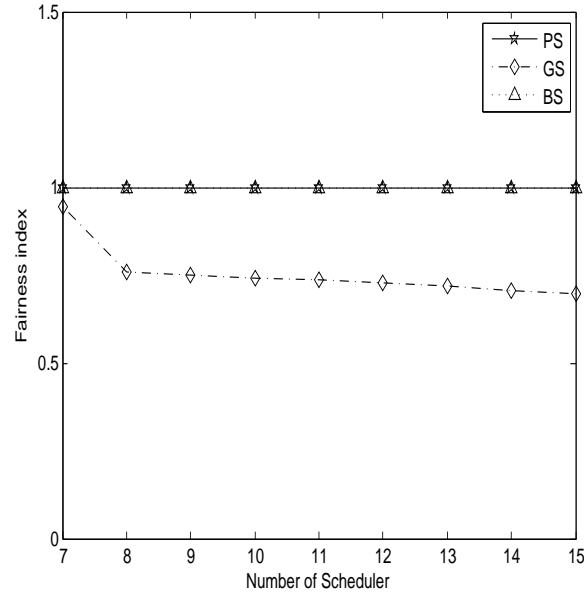


Fig. 17. Fairness versus Number of Scheduler

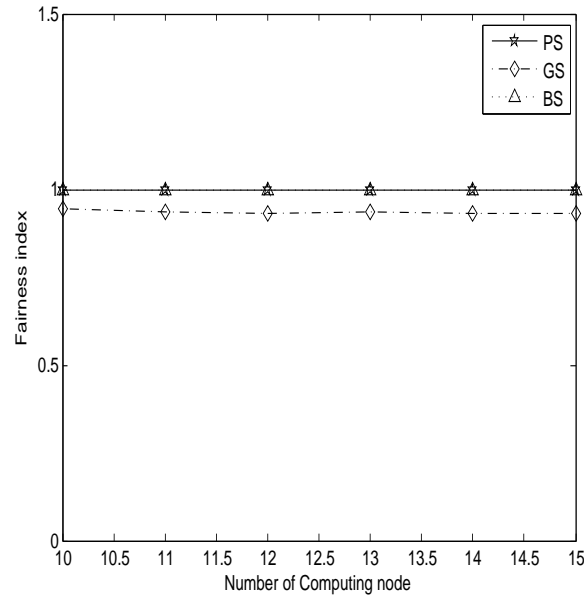


Fig. 18. Fairness versus Number of Computing Node

- [5] L. MNI, K. HWANG. *Optimal Load Balancing in a Multiple Processing System with Many Job Classes*, IEEE Transactions on software engineering, 1985, SE-11(5):491–496.
- [6] J. LI, H. KAMEDA. *Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems*, IEEE Transactions on computers, 1998, 47(3):322–332.
- [7] A. G. DELAVAR, M. NEJADKHEIRALLAH, M. MOTALLEB. *A New Scheduling Algorithm for Dynamic Task and Fault Tolerant in Heterogeneous Grid Systems Using Genetic Algorithm*, In: Proc. of the IEEE International conference on computer science and information technology, 2010, 9:408–412.
- [8] N. FUJIMOTO, K. HAGIHARA. *Near-optimal Dynamic Task Scheduling of Independent Coarse-grained Tasks onto a Computational Grid*, In: Proc. of the 2003 international conference on parallel processing, 2003:391–398.

- [9] R. MAHAJAN, M. RODRIG, D. WETHERALL, et al. *Experiences Applying Game Theory to System Design*, In: Proc. of the 2004 Annual Conference of The Special Interest Group on Data Communication [C], Portland, Oregon, USA:ACM Press, 2004:183–190.
- [10] K. RANGANATHAN, M. RIPEANU, A. SARIN, et al. *Incentive Mechanisms for Large Collaborative Resource Sharing*, In: Proc. of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid [C], Chicago, Illinois, USA: IEEE Computer Society, 2004:1–8.
- [11] D. GROSU, A. T. CHRONOPOULOS. *Non-cooperative Load Balancing in Distributed Systems*, Journal of Parallel and Distributed Computing, 2005, 65:1022–1034.
- [12] K. YI, R. WANG. *Nash Equilibrium Based task Scheduling Algorithm of Multi-schedulers in Grid Computing*, ACTA Electronica Sinica, 2009, 37(2):329–333.
- [13] R. SUBRATA, A. Y. ZOMAYA, B. LANDFELDT. *Game Theoretic Approach for Load Balancing in Computational Grids*, IEEE Transactions on Parallel and Distributed Systems, 2008, 19(1):66–76.
- [14] G. WEI, A. V. VASILAKOS, N. XIONG. *Scheduling Parallel Cloud Computing Services: An Evolutional Game*, In: Proc. of the 1st International Conference on Information Science and Engineering(ICISE2009), 2009:376–379.
- [15] O.M. Elzeki, M. Z. Reshad, M. A. Elsoud. *Improved Max-Min Algorithm in Cloud Computing*, International Journal of Computer Applications, 2012, 50.
- [16] G. LIU, J. LI, J. XU. *An Improved Min-min Algorithm in Cloud Computing*, In: Proc. of the 2012 International Conference of Modern Computer Science and Applications, 2013, 191:47–52.
- [17] C Zhao, S Zhang, Q Liu. *Independent tasks scheduling based on genetic algorithm in cloud computing*, Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on. IEEE, 2009: 1-4.
- [18] K. Li, G. Xu, G. Zhao. *Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization*, Chinagrid Conference (ChinaGrid), 2011 Sixth Annual. IEEE, 2011: 3-9.
- [19] S. S.CHANHAN, R. JOSHI. *A Heuristic for QoS Based Independent Task Scheduling in Grid Environment*, In: Proc. of the International Conference on Industrial and Information system. 2010:102–106.
- [20] M. XU, L. CUI, H. WANG, Y. BI. *A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing*, In: Proc. of the 2009 IEEE International Symposium on Parallel and Distributed Processing with Application,2009:629–634.
- [21] E. DEELMAN, G. SINGH, M. LIVNY, et al *The Cost of Doing Science on the Cloud: the Montage Example*, In: Proc. of the ACM/IEEE Conference on SupercomputingPiscataway:IEEE Press, 2008:1–12.
- [22] M. ASSUNCAO, A. COSTANZO, R. BUYYA. *Evaluating the Cost-benefit of Using Cloud Computing to Extend the Capacity of Clusters*, In: Proc. of the 18th ACM International Symposium on High Performance Distributed ComputingNew York: ACM Press2009:141–150
- [23] G. TIAN, D. MENG, J. ZHAN. *Reliable Resource Provision Policy for Cloud Computing*, Chinese Journal of Computers. 2010, 33(10):1859–1872.
- [24] R. SUN, J. LI. *The Basis of Queue Theory*, Beijing, China: Science Publisher, 2002.
- [25] H. Tang,X. Qin. *Practical Optimization Methods*, China: Dalian University of Technology Press. 2005.
- [26] Q. Yang. *The Entropy Function Methods For Solving Minimax Problems*, Acta Scientiarum Naturalium Universitatis Nankaiensis, 2001, 34(3):7–15.
- [27] Y. CHOW, W. KOHLER. *Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System*, IEEE Transactions on Computers, 1979, 28:354–361.